Camel is an integration framework that aims to make your integration projects productive and fun. The Camel project was started in early 2007, but although it’s relatively young, Camel is already a mature open source project, available under the liberal Apache 2 license, and it has a strong community.

Camel’s focus is on simplifying integration. We’re confident that by the time you finish reading these pages, you’ll appreciate Camel and add it to your “must have” list of tools

**Why do we need integration?**

Enterprises are typically comprised of hundreds if not thousands of applications that are custom-built, acquired from a third-party, part of a legacy system, or a combination thereof, operating in multiple tiers of different operating system platforms.

First of all, writing business applications is hard. Creating a single, big application to run a complete business is next to impossible.

Second, spreading business functions across multiple applications

provides the business with the flexibility to select the “best” accounting package, the “best” customer relationship management or the order processing system that best suits the business’ needs

In order to support common business processes and data sharing across applications, these applications need to be integrated. Application integration needs to provide efficient, reliable and secure data exchange between multiple enterprise applications.

EIP BOOK – Gregor Hohpe and Bobby Woolf

ROUTING AND MEDIATION ENGINE

The core feature of Camel is its routing and mediation engine.

DOMAIN-SPECIFIC LANGUAGE (DSL)

Camel’s domain-specific language (DSL) is a major contribution to the integration

space. A few other integration frameworks currently feature a DSL (and some allow

you to use XML to describe routing rules), but unlike Camel their DSLs are based on

custom languages. Camel is unique because it offers multiple DSLs in regular programming

languages such as Java, Scala, Groovy, and it also allows routing rules to be

specified in XML.

EXTENSIVE COMPONENT LIBRARY

Camel provides an extensive library of more than 80 components. These components

enable Camel to connect over transports, use APIs, and understand data formats.

PAYLOAD-AGNOSTIC ROUTER

Camel can route any kind of payload—you aren’t restricted to carrying XML payloads.

This freedom means that you don’t have to transform your payload into a canonical

format to facilitate routing.

PAYLOAD-AGNOSTIC ROUTER

Camel can route any kind of payload—you aren’t restricted to carrying XML payloads.

This freedom means that you don’t have to transform your payload into a canonical

format to facilitate routing.

EASY CONFIGURATION

The convention over configuration paradigm is followed whenever possible, which minimizes

configuration requirements. In order to configure endpoints directly in routes,

Camel uses an easy and intuitive URI configuration.

For example, you could configure a file consumer to scan recursively in a subfolder

and include only a .txt file, as follows:

from("file:data/inbox?recursive=true&include=\*.txt")

AUTOMATIC TYPE CONVERTERS

Camel has a built-in type-converter mechanism that ships with more than 150 converters.

You no longer need to configure type-converter rules to go from byte arrays to

strings, for example. And if you find a need to convert to types that Camel doesn’t support,

you can create your own type converter. The best part is that it works under the

hood, so you don’t have to worry about it.

LIGHTWEIGHT CORE

Camel’s core can be considered pretty lightweight, with the total library coming in at

about 1.6 MB and only having a dependency on Apache Commons Logging and Fuse-

Source Commons Management. This makes Camel easy to embed or deploy anywhere

you like, such as in a standalone application, web application, Spring application, JavaEE application, JBI container, OSG i bundle, Java Web Start, or on the Google App

engine

TEST KIT

Camel provides a Test Kit that makes it easier for you to test your own Camel applications.